BÀI 9: SỰ PHÂN LỚP DỮ LIỆU

I. Mục tiêu:

Sau khi thực hành xong, sinh viên nắm được sự phân lớp dữ liệu thông qua:

- Support Vector Machines

II. Tóm tắt lý thuyết:

Support Vector Machines (SVM) được xác định cho sự phân lớp nhị phân của dữ liệu số. Bài toán lớp nhị phân có thể được tổng quát hóa thành trường hợp nhiều lớp.
Giả sử rằng các nhãn của lớp được vẽ từ $\{-1, 1\}$. Với tất cả các mô hình tuyến tính, SVM sử dụng việc tách các siêu phẳng (hyperplanes) như biên quyết định (decision boundary) giữa 2 lớp. Trong trường hợp của SVM, bài toán tối ưu của việc xác định các siêu phẳng này được ký hiệu là *lề* (margin). Một *siêu phẳng lề lớn nhất* (maximum margin hyperplane) tách dữ liệu thành 2 lớp.
Xây dựng các siêu phẳng song song mà chúng tiếp xúc với dữ liệu huấn luyện của các lớp đối diện ở cả 2 bên, và không có điểm dữ liệu giữa chúng. Các điểm dữ liệu huấn luyện trong các siêu phẳng này được gọi là *các vector hỗ trợ* (support vectors), và khoảng cách giữa hai siêu phẳng là lề (margin).
Cho n điểm dữ liệu trong tập huấn luyện $\mathcal{D}$ được ký hiệu bởi $(\overline{X_1}, y_1) \dots (\overline{X_n}, y_n)$, với $\overline{X_i}$ là vector dòng d chiều tương ứng với điểm dữ liệu thứ i, và $y_i \in \{-1, +1\}$ là biến của lớp nhị phân của điểm dữ liệu thứ i. Khi đó, siêu phẳng tách rời có dạng như sau:

$$\overline{W} \cdot \overline{X} + b = 0$$

với $\overline{W} = (w_1 \dots w_d)$ là vector dòng d chiều tương ứng với phương trực giao với siêu phẳng, và b là một vô hướng, cũng được biết như là nhiễu (bias). Vector $\overline{W}$ điều chỉnh sự định hướng của siêu phẳng và nhiễu b điều chỉnh khoảng cách của siêu phẳng từ gốc. Do đó, 2 siêu phẳng tách rời có thể được biểu diễn trong dạng theo sau:

$$\overline{W} \cdot \overline{X} + b = +1$$

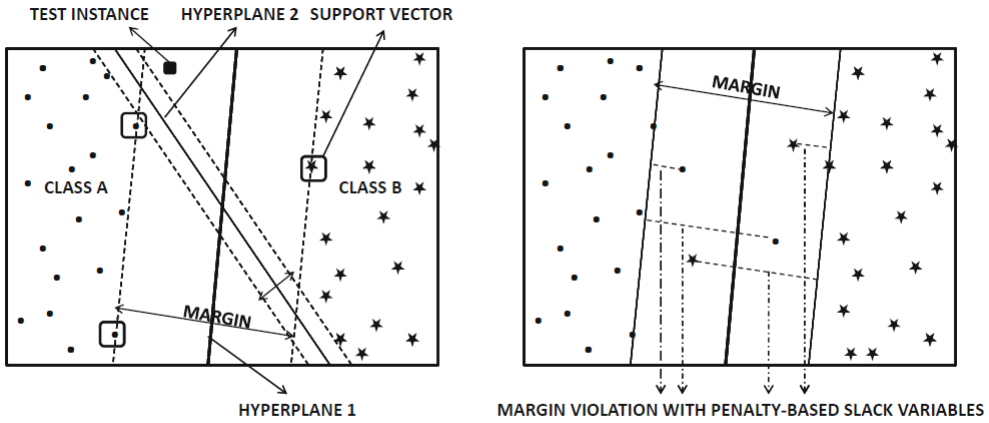
$$\overline{W} \cdot \overline{X} + b = -1$$

Các ràng buộc này liên quan tới *các ràng buộc lề*. Hai siêu phẳng phân đoạn không gian dữ liệu thành 2 miền (vùng). Giả sử rằng không điểm dữ liệu huấn luyện nào nằm trong miền biên quyết định không chắn chắn giữa 2 siêu phẳng này, và tất cả các điểm dữ liệu huấn luyện cho mỗi lớp được ánh xạ thành một trong số 2 miền cực trị. Điều này có thể được biểu diễn như các ràng buộc theo từng điểm trong các điểm dữ liệu huấn luyện như sau:

$$\overline{W} \cdot \overline{X_i} + b \geqslant +1, \quad \forall i : y_i = +1$$
$$\overline{W} \cdot \overline{X_i} + b \leqslant -1, \quad \forall i : y_i = -1$$

Với trường hợp dữ liệu không tách rời, mức độ vi phạm của mỗi ràng buộc lề bằng việc huấn luyện điểm dữ liệu $\overline{X_i}$ được ký hiệu bởi một biến yếu (slack) $\xi_i \geqslant 0$. Do đó, tập hợp các ràng buộc mềm mới trong các siêu phẳng tách rời có thể được biểu diễn như sau:

$$\overline{W} \cdot \overline{X_i} + b \geqslant +1 - \xi_i, \quad \forall i : y_i = +1$$
$$\overline{W} \cdot \overline{X_i} + b \geqslant -1 + \xi_i, \quad \forall i : y_i = -1$$
$$\xi_i \geqslant 0 \quad \forall i.$$



(a) Hard separation (b) Soft separation

III. Nội dung thực hành:

1. SVM với tập dữ liệu tách rời tuyến tính

- Cài đặt ipywidgets

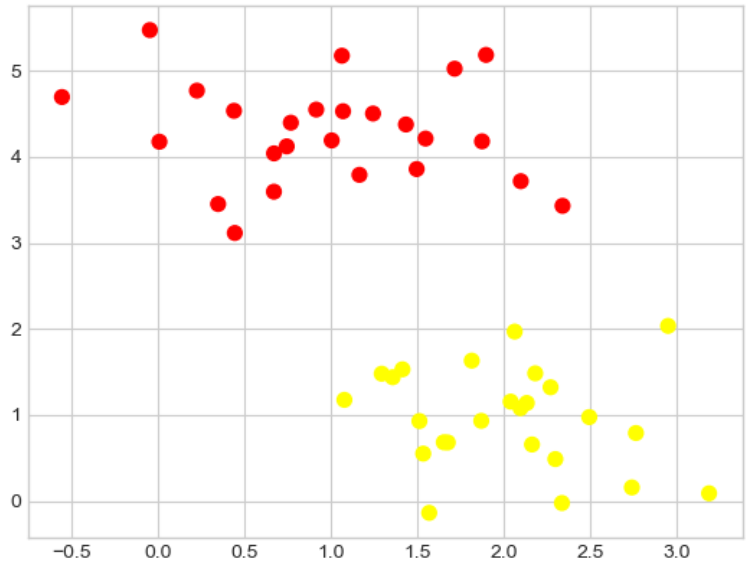
```
Microsoft Windows [Version 10.0.19045.2965]
(c) Microsoft Corporation. All rights reserved.

C:\Users\Huynh>pip install ipywidgets
Collecting ipywidgets
  Downloading ipywidgets-8.0.6-py3-none-any.whl (138 kB)
     ---------------------------------------- 138.3/138.3 kB 2.1 MB/s eta 0:00:00
Collecting ipykernel>=4.5.1
  Downloading ipykernel-6.16.2-py3-none-any.whl (138 kB)
     ---------------------------------------- 138.5/138.5 kB 4.1 MB/s eta 0:00:00
Collecting traitlets>=4.3.1
  Downloading traitlets-5.9.0-py3-none-any.whl (117 kB)
     ---------------------------------------- 117.4/117.4 kB 6.7 MB/s eta 0:00:00
Collecting jupyterlab-widgets~=3.0.7
  Downloading jupyterlab_widgets-3.0.7-py3-none-any.whl (198 kB)
     ---------------------------------------- 198.2/198.2 kB 6.1 MB/s eta 0:00:00
Collecting ipython>=6.1.0
  Downloading ipython-7.34.0-py3-none-any.whl (793 kB)
     ---------------------------------------- 793.8/793.8 kB 3.6 MB/s eta 0:00:00
Collecting widgetsnbextension~=4.0.7
  Downloading widgetsnbextension-4.0.7-py3-none-any.whl (2.1 MB)
     ---------------------------------------- 2.1/2.1 MB 4.5 MB/s eta 0:00:00
Collecting debugpy>=1.0
  Downloading debugpy-1.6.7-cp37-cp37m-win_amd64.whl (4.8 MB)
     ---------------------------------------- 4.8/4.8 MB 5.3 MB/s eta 0:00:00
```

- Import thư viện

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC # "Support vector classifier"
plt.style.use('seaborn-whitegrid')
from scipy import stats
from ipywidgets import interact, fixed
from sklearn.datasets import make_blobs
```

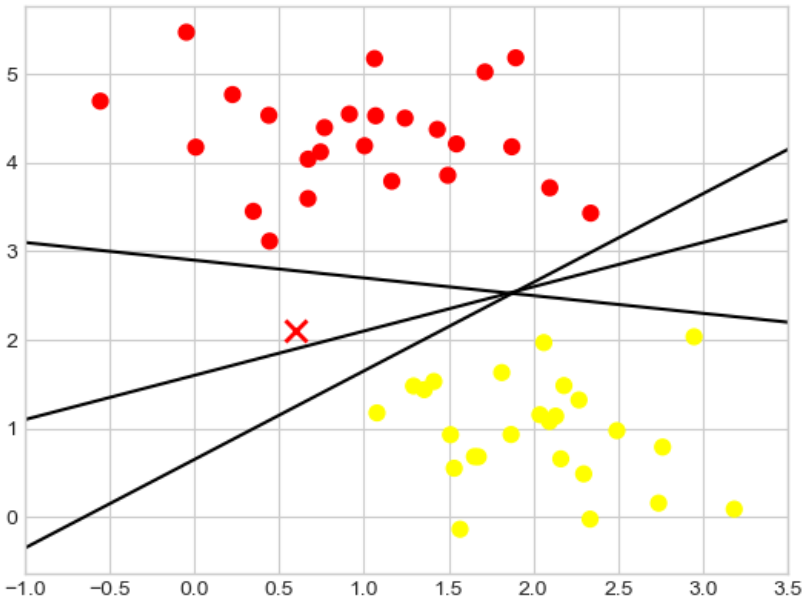
- Khởi tạo tập dữ liệu có 50 điểm và 2 cụm

```
X, y = make_blobs(n_samples=50, centers=2,
                  random_state=0, cluster_std=0.60)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn');
plt.show()
```



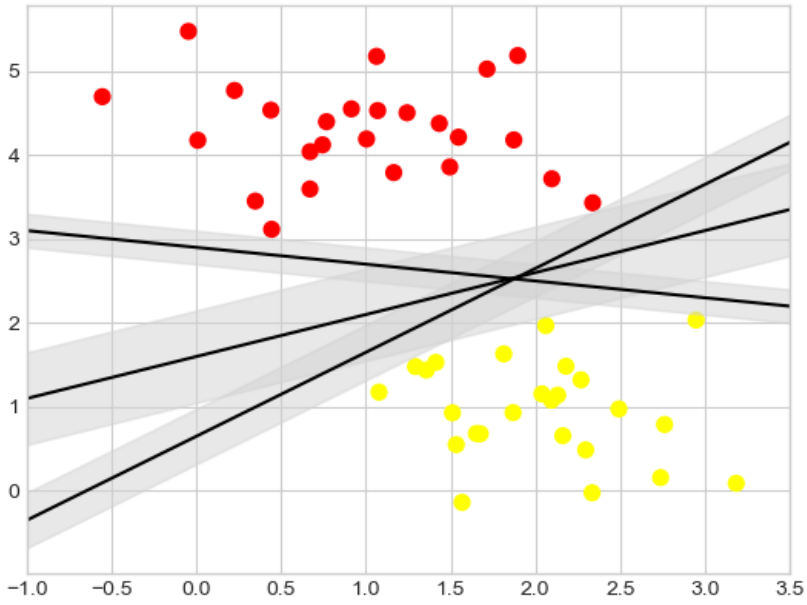
- Vẽ đường thẳng phân tách 2 cụm

```
xfit = np.linspace(-1, 3.5)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')
plt.plot([0.6], [2.1], 'x', color='red', markeredgewidth=2, markersize=10)

for m, b in [(1, 0.65), (0.5, 1.6), (-0.2, 2.9)]:
    plt.plot(xfit, m * xfit + b, '-k')

plt.xlim(-1, 3.5);
plt.show()
```



- Vẽ siêu phẳng lề lớn nhất

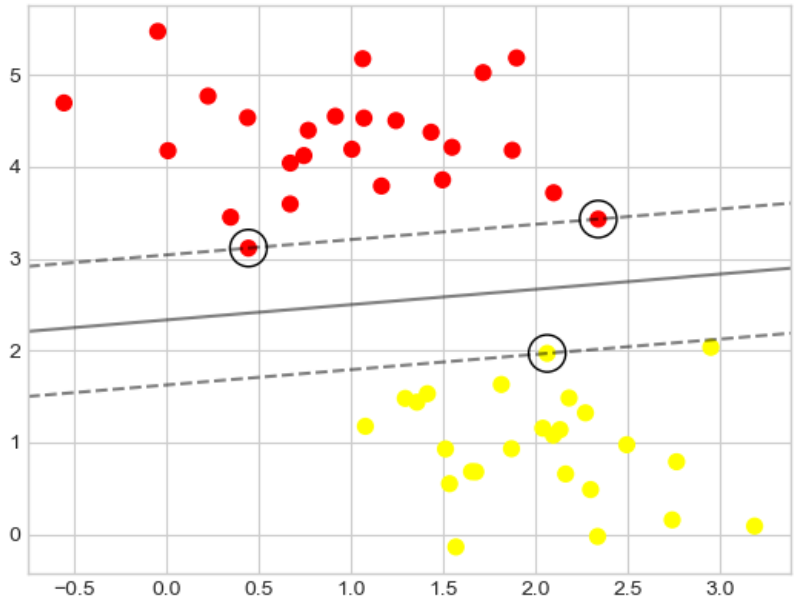
```
xfit = np.linspace(-1, 3.5)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')

for m, b, d in [(1, 0.65, 0.33), (0.5, 1.6, 0.55), (-0.2, 2.9, 0.2)]:
    yfit = m * xfit + b
    plt.plot(xfit, yfit, '-k')
    plt.fill_between(xfit, yfit - d, yfit + d, edgecolor='none',
                     color='lightgray', alpha=0.5)

plt.xlim(-1, 3.5);
plt.show()
```



- Sử dụng support vector classifier (SVC) của Scikit-Learn để huấn luyện mô hình SVM và vẽ các biên quyết định của SVM

```
model = SVC(kernel='linear', C=1E10)
model.fit(X, y)
def plot_svc_decision_function(model, ax=None, plot_support=True):
    """Plot the decision function for a 2D SVC"""
    if ax is None:
        ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # create grid to evaluate model
    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)
    Y, X = np.meshgrid(y, x)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
    P = model.decision_function(xy).reshape(X.shape)

    # plot decision boundary and margins
    ax.contour(X, Y, P, colors='k',
               levels=[-1, 0, 1], alpha=0.5,
               linestyles=['--', '-', '--'])

    # plot support vectors
    if plot_support:
        ax.scatter(model.support_vectors_[:, 0],
                   model.support_vectors_[:, 1],
                   s=300, linewidth=1, edgecolors='black',
                   facecolors='none');
    ax.set_xlim(xlim)
    ax.set_ylim(ylim)


plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')
plot_svc_decision_function(model);
plt.show()
model.support_vectors_
```

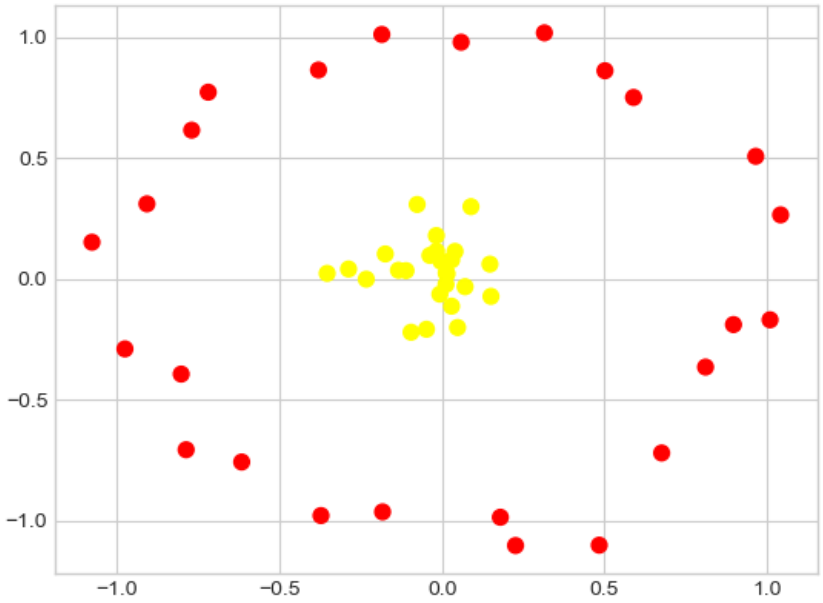
```
array([[0.44359863, 3.11530945],
       [2.33812285, 3.43116792],
       [2.06156753, 1.96918596]])
```



2. SVM với biên quyết định không tuyến tính

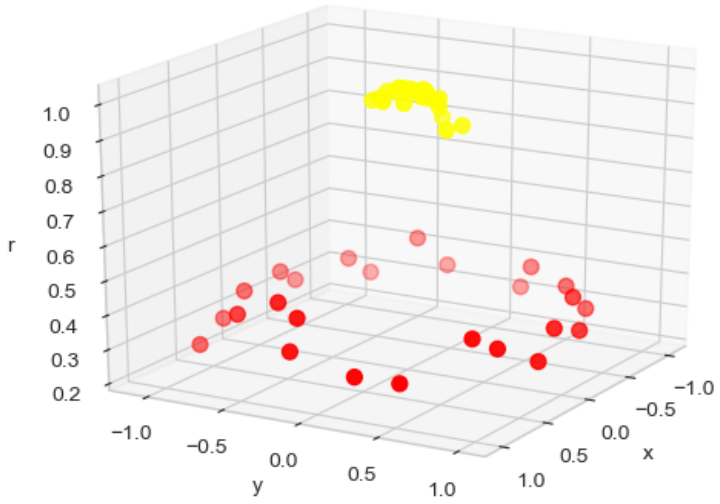
- Khởi tạo tập dữ liệu có 50 điểm.

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.svm import SVC # "Support vector classifier"
plt.style.use('seaborn-whitegrid')
from scipy import stats
from ipywidgets import interact, fixed
from sklearn.datasets import make_blobs, make_circles
from mpl_toolkits import mplot3d

X, y = make_circles(50, factor=.1, noise=.1)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn');
plt.show()
```



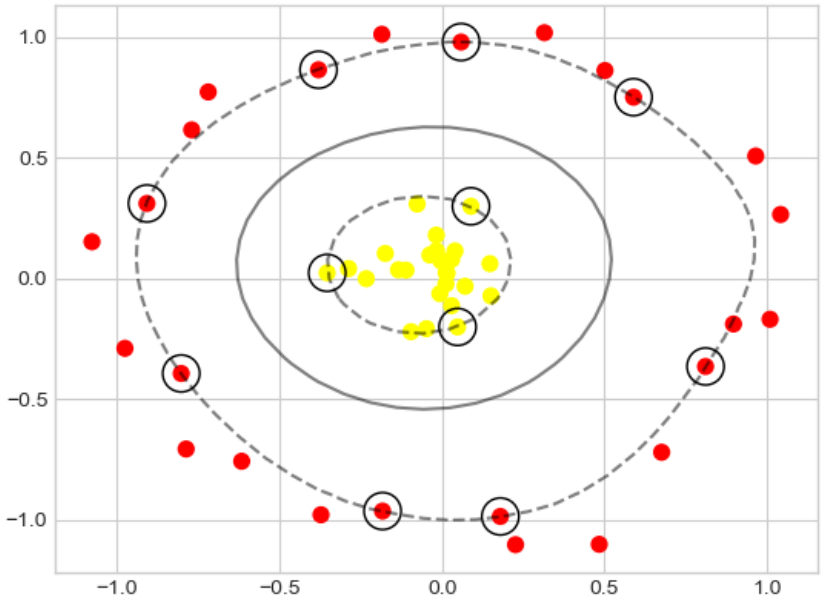
- Sử dụng hàm radial basis function và vẽ dữ liệu trong không gian 3 chiều

```
xfit = np.linspace(-1, 3.5)
plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')
plt.plot([0.6], [2.1], 'x', color='red', markeredgewidth=2, markersize=10)

for m, b in [(1, 0.65), (0.5, 1.6), (-0.2, 2.9)]:
    plt.plot(xfit, m * xfit + b, '-k')

plt.xlim(-1, 3.5);
plt.show()
```



- Sử dụng support vector classifier (SVC) của Scikit-Learn để huấn luyện mô hình SVM và vẽ các biên quyết định của SVM

```
model = SVC(kernel='rbf', C=1E6)
model.fit(X, y)
def plot_svc_decision_function(model, ax=None, plot_support=True):
    """Plot the decision function for a 2D SVC"""
    if ax is None:
        ax = plt.gca()
    xlim = ax.get_xlim()
    ylim = ax.get_ylim()

    # create grid to evaluate model
    x = np.linspace(xlim[0], xlim[1], 30)
    y = np.linspace(ylim[0], ylim[1], 30)
    Y, X = np.meshgrid(y, x)
    xy = np.vstack([X.ravel(), Y.ravel()]).T
    P = model.decision_function(xy).reshape(X.shape)

    # plot decision boundary and margins
    ax.contour(X, Y, P, colors='k',
               levels=[-1, 0, 1], alpha=0.5,
               linestyles=['--', '-', '--'])

    # plot support vectors
    if plot_support:
        ax.scatter(model.support_vectors_[:, 0],
                   model.support_vectors_[:, 1],
                   s=300, linewidth=1, edgecolors='black',
                   facecolors='none');
    ax.set_xlim(xlim)
    ax.set_ylim(ylim)


plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='autumn')
plot_svc_decision_function(model);
plt.show()
model.support_vectors_
```



3. Yêu cầu:

- Áp dụng tương tự với tập dữ liệu banknote authentication từ UCI Machine Learning Repository.
- Viết file báo cáo.